第57回 プログラミング･シンポジウム 2016.1.8-10

有限体積法を高速化するための領域特化言語のC++への埋め込み

伊藤 正勝[1,a)]　宮島 敬明[1]　藤田 直行[1]

概要：我々は、数値流体力学のシミュレーション開発において、ソースコードを書くために、モデルの数式を分解して冗長な表現に変換せざるをえないことが、開発と高速化の妨げになっていると考えている。
そこで、この妨げを減らすために、領域特化言語を設計してC++に埋め込んでいる。我々の領域特化言語は、有限体積法の数式をハードウェアに近い低レベルのコードに変換してC++コンパイラに渡すので、開発者は、シミュレーションモデルの数式を直接的にソースコードにすることができる。今回は、流体力学の重要現象である拡散と対流のうち、拡散現象を扱えるように、試作版を作成した。
今後は、もう一方の重要現象である対流を扱えるように機能を拡張し、バックエンドに、自動並列化とチューニング機能を追加する予定である。

キーワード：領域特化言語、高速化、有限体積法、C++、Boost.Proto

1. はじめに

1957年にFORTRANが提供されてから現在に至るまで、科学シミュレーション開発者は常にプログラミング言語の問題に悩まされてきた。しかも、FORTRAN77の代わりにFortran90/95やC++が使えるようになってからも、問題はなくなってはいない。ただ、最近では、計算機科学の専門家でなくとも領域特化言語(DSL)[1]を試作できるようになったため[2]、我々は各自の専門領域でDSLを作成するという自由も持っている。

我々の専門領域は、数値流体力学(CFD)シミュレーションのスパコン・並列計算機での計算速度向上である。現在、速度向上のためには、ハードに近い低レベルの記述とシミュレーションモデルの記述の間でかなりの摺り合わせが必要である。それで、この摺り合わせが、CFDアプリ開発者からシミュレーションモデルについて考える時間を奪っており、この問題を低減したいと思っている。

理想は、DSLコンパイラに、シミュレーションモデルの記述である数式を、自動的にチューニングされ並列化されたコードへと変換させることである。このようなDSLが実現できれば、CFDアプリ開発者はシミュレーションモデルの数式を直接的にソースコードにすることができるので、現在よりももっと長い時間をモデルについて考えるために使うことができる。

1.1 埋め込み型のDSL

DSLのタイプは、我々にとっては、Fortanのよ

[1] 宇宙航空研究開発機構 航空技術部門 数値解析技術研究ユニット
〒182-8522 調布市深大寺東町 7-44-1
[a)] itoh.masakatsu@jaxa.jp

うな独立型よりも、C++をホスト言語とする埋め込み型のほうが望ましい。これはCFD分野でオープンソースのフレームワークとして公開されている OpenFOAM[3] や SU^2[4] のソースコードを見てみると、この分野でも、オブジェクト指向型プログラミングや、テンプレートメタプログラミング、(現状では不完全だが関数型プログラミング)のような言語機能が役立っているように思えるからである。

1.2 数式テンプレートに基づく DSL

しかし、C++には悩ましい問題もある。それはソースコードをモデルの数式の記法に近づけるために、モデルの構成概念をオブジェクト化すると、深刻な速度低下が起こることがあるという問題である。ソースコードを線形代数の数式の記法に近づけるために、行列やベクトルをオブジェクト化すると、それらの一時オブジェクトの生成とコピーが計算時間が数十倍に長くしてしまう。これは、大規模計算の必要な CFD 分野では受け入れがたいことである。

ただ、この問題は数式テンプレート技法 [5] で回避でき、しかも、数式テンプレートをベースとする DSL の開発例 [6][7][8][9] も増えつつある。

こうして見ていくと、残る課題は、数式テンプレートと DSL のギャップを埋めることである。この課題に対しては、我々は DSL の意味論モデル [1] が鍵になると考えている。そこで、CFD 分野で広く使われている有限体積法 [10] のための意味論モデルを提案すべく、DSL としての数式テンプレートライブラリ、「有限体積法テンプレート」を開発している。

本稿では、第 2 節で、意味論モデルについて説明し、第 3 節では、意味論モデルがサポートするオブジェクトモデルについて述べつつ、有限体積法テンプレートの使用法を説明する。

2. 有限体積法テンプレートの意味論モデル

意味論モデルとは、埋め込み型の DSL においては、ソースコードから低レベルのコードへの変換を規定するものである。他方で、独立型の DSL においては、意味論モデルはソースコードの評価の仕方を規定する。これらを一般化すると、ソースコードが何をするかを規定するのが意味論モデルである。

2.1 科学シミュレーション分野での意味論モデル

意味論モデルは、DSL のバックエンドに隠れており、フロントエンドのオブジェクトモデルに比べると目立たないが、科学シミュレーション分野では重要な働きをしている。それは、オブジェクトモデルで規定しづらい数式の変換を規定することである。例えば、構造解析分野では、有限要素法モデリングのための DSL[6] が開発されており、その DSL が埋め込まれた C++コンパイラはモデリングの数式を実行形式のコードへと変換している。ここで、DSL の意味論モデルは、有限要素法の数式の抽象度を落とし、冗長な表現の数式コードに変換して、C++コンパイラに渡している。同様に、我々の提案する意味論モデルは、有限体積法の数式の抽象度を落として、C++コンパイラに渡すためのものである。

2.2 意味論モデルが行うべき数式変換

今回は、意味論モデルが扱う範囲を、CFD 分野で重要な現象、対流と拡散のうちで、拡散現象に絞ることにした。

本稿では、拡散現象の一種である定常熱伝導問題を題材として、有限体積法の数式変換を説明する。これらの数式変換を DSL の意味論モデルとして記述すれば、バックエンドで動かすことができるわけである。

解析的に解ける題材として、空気中に置かれた円柱での熱伝導を考える。単純化のために、円柱の断面積方向の温度分布は無視して、温度分布は $T(x)$ で表現されるとする。周囲の空気の温度を T_∞ とすると、この問題の微分方程式は

$$kA\frac{d^2}{dx^2}T(x) - h\,P\,(T(x) - T_\infty) = 0 \qquad (1)$$

で与えられる。第 1 項は熱拡散、第 2 項は周囲の空気の対流による冷却あるいは加熱速度を表す。

ここで、円柱の断面積を A、周囲の長さ P、熱伝導係数 k、対流伝熱伝達係数 h としている。

そして、温度分布を決定する境界条件として、円柱の左端は温度一定の高熱源に接していて、右端は断熱されているとする。高熱源の温度を T_B、左端から右端までの長さを L とすると、これらの境界条件は、

$$T(0) = T_B \tag{2}$$

$$\frac{d}{dx}T(L) = 0 \tag{3}$$

と表現される。

すると、微分方程式の解を求めることができて、

$$T(x) = T_\infty + (T_B - T_\infty)\frac{\cosh(n(L-x)))}{\cosh(nL)}$$

となる。ここで、T_B は高温板の温度、L は円柱の長さ、$n^2 = hP/(\lambda A)$ である。

さて、ここからは、この問題を有限体積法で解くために必要な数式変換をたどっていく。まず、有限体積法のモデルでは、温度を $T(x)$ のような連続分布から、格子点 x_i の上の温度 T_i として、離散データに変換する。換言すれば、連続関数 $T(x)$ を、離散化して、i 番目の要素が T_i であるようなベクトル $\mathbf{t}$ に変換したわけである。次に、微分方程式のなかの演算子

$$kA\frac{d^2}{dx^2} - h\,P\,\hat{I}_d \tag{4}$$

も離散化して、行列 C として表現できる。ここで、$\hat{I}_d$ は恒等演算子で、任意の関数 $f(x)$ に作用させた時に $\hat{I}_d f(x) = f(x)$ となるような演算子である。さらに、演算子にも温度分布にも関係のない項も、ベクトル $\mathbf{b}$ に変換する。

こうして、離散化によって微分方程式は

$$C\,\mathbf{t} = \mathbf{b} \tag{5}$$

のような行列方程式（連立方程式）に変換される。あとは、この方程式を境界条件 (2)、(3) のもとで補正して、数値的に解けば、温度分布 $\mathbf{t}$ が求まる。

つまり、我々の意味論モデルがすべき数式変換とは、微分方程式の演算子と関数を、行列とベクトルに変換することである。

2.3 線型代数の意味論モデル

ただし、行列方程式 (5) を数値解法を行列オブジェクトとベクトルオブジェクトの数式を使って記述するために、数式テンプレートが必要である。

ここで、数式テンプレートだけを使って計算速度の低下を防ぐこともできるが、我々は数式テンプレート実装の煩雑さを低減するために、線形代数の意味論モデルを作成した。

このために、我々は数式テンプレートをベースとする DSL の試作を支援する Boost.Proto ライブラリ [2] を使用した。

2.4 有限体積法の意味論モデル

Boost.Proto を使うと、意味論モデルによる数式テンプレートの変換を、テンプレートメタプログラミングで記述することができる。

二階微分演算子 d^2/dx^2 のための意味論モデルを以下に示す。この意味論モデルは、d^2/dx^2 に対応する数式テンプレートにパターンマッチして、遅延評価で行列オブジェクトを生成する閉包オブジェクトに置き換えるためのものである。その定義は、Boost.Proto の記法で次のようになる。

```
struct SecondDiffOprType : proto::
    or_<
  proto::when<
    proto::multiplies<
      proto::terminal< DiffOpr >,
      proto::terminal< DiffOpr > >,
      proto::make_terminal(
        SecondDiffQuotinent() )
  >,
  proto::when<
    proto::multiplies<
      proto::multiplies<
        proto::terminal< proto::_ >,
        proto::terminal< DiffOpr >
            >,
      proto::terminal< DiffOpr >
    >,
    proto::make_multiplies<
      DiffOprFactor( proto::_left),
```

```
      proto::make_terminal(
        SecondDiffQuotinent() )
    >
  >,
  proto::plus< SecondDiffOprType,
               SecondDiffOprType >,
  proto::minus< SecondDiffOprType,
                SecondDiffOprType >
> {};
```

コード 1 : 二階微分演算子を変換する意味論モデル

ここで、テンプレートパラメターに現れる DiffOpr クラスは、有限体積法テンプレートでは、一階の微分演算子 d/dx の型を表現するクラスである。そのインスタンス diffOpr と実数型の任意の数式コード a について、この意味論モデルは、パターン diffOpr*diffOpr、 a*diffOpr*diffOpr と、これらの和と差で表現されるようなパターンの数式コードにマッチする。そして、マッチした数式テンプレートのなかの二階微分演算子を、閉包オブジェクト SecondDiffQuotinent に置き換える。この SecondQuotinent オブジェクトは遅延評価されるときに、行列オブジェクトを生成する。

数式で表現すると、この意味論モデルと遅延評価によって、二階微分演算子は次のように行列へと離散化される。

$$\frac{d^2}{dx^2} \rightarrow \begin{pmatrix} \ddots & 1/\Delta x & & \\ 1/\Delta x & -2/\Delta x & 1/\Delta x & \\ & 1/\Delta x & -2/\Delta x & 1/\Delta x \\ & & 1/\Delta x & \ddots \end{pmatrix}$$

ここで、Δx は隣接する格子点 x_i, x_{i+1} の間隔である。

また、同様に、関数からベクトルへの離散化も、意味論モデルによる数式テンプレートの変換と遅延評価によって実現できる。

3. 有限体積法テンプレートの使い方

意味論モデルは DSL のバックエンドに隠れていて、DSL ユーザーから使われることはないのに対して、オブジェクトモデルはフロントエンドで DSL ユーザーとのインタフェースを提供している。

フロントエンドに属するオブジェクトは、３つの名前空間に分かれている。

(1) FVM 名前空間
- Grid オブジェクト – シミュレーション領域を分割
- BoundaryCorrector オブジェクト – 境界条件により行列とベクトルを補正する。

(2) DLA 名前空間
- Matrix オブジェクト、Vector オブジェクト

(3) SLA 名前空間
- ConjugateGradient オブジェクト– 行列方程式の反復解法
- DiagonalPreconditioner オブジェクト– 前処理

これらの名前空間は、有限体積法、行列・ベクトル、反復解法のためのものである。

3.1 熱伝導問題を解くためのコード

以下で、2.2 節の一次元定常熱伝導問題を解くために、これらのオブジェクトをどのように使用するかを示す。

```
FVM::Grid grid( N, L);
grid.addDirichletBoundary(
    -1, 0, TB);
grid.NeumannBoundary(
     N, N-1, 0.0);
```

コード 2 : シミュレーション領域の設定

まず、Grid オブジェクトをインスタンス化する際には、シミュレーション領域の分割数 N と長さ L を指定する。境界条件を追加する際には、引数として、境界の外側の格子点番号、内側の格子点番号、境界で固定する値、を指定する。

```
auto opr = proto::deep_copy(
  k * A *
    FVM::diffOpr * FVM::diffOpr
  - h * P * FVM::identityOpr );

FVM::BoundaryCorrector
```

```
  bCorrector( grid, opr);
```

コード 3 : 境界条件による補正オブジェクト

ここで、変数 opr には、(4) 式の演算子に対応する数式テンプレートのコピーを格納している。ここで微分演算子を指定するために有限体積法テンプレートが提供する静的オブジェクト FVM::diffOpr を使っている。演算子 opr と Gird インスタンスから補正オブジェクトをインスタンス化する。

```
DLA::Matrix C =
  grid.discretizeOperator( opr );
DLA::Vector b =
  grid.discretizeFunction(
            h * P * TI );
DLA::Vector guess =
  grid.discretizeFunction(
            ( TI + TB ) / 2.0 );
```

コード 4 : 行列方程式

行列方程式の係数行列と定数項ベクトルは、Grid インスタンスに演算子 opr、定数項の値 hPT_∞ を渡して、生成させる。また、温度分布の初期推定値 guess もここで生成させる。

```
bCorrector.applyTo( C);
bCorrector.applyTo( b);
```

コード 5 : 境界条件による補正

係数行列と定数項ベクトルをそれぞれに境界条件で補正する。

```
SLA::DiagonalPreconditioner
  precond( C);
SLA::ConjugateGradient< DLA::Matrix,
      SLA::DiagonalPreconditioner >
   cg( C, precond);

const double eps = 1.0e-5;
DLA::Vector t =
  cg.solve( b, guess, eps);
```

コード 6 : 行列方程式の復解法

前処理オブジェクトを係数行列からインスタンス化して、さらに、共役勾配法オブジェクトをインスタンス化する。最後に、収束条件 eps を指定して、共役勾配法で解く。こうして、温度分布がベクトルオブジェクト t として求まり、定常熱伝導問題の答えが得られる。

以上のように DSL を使うことで、有限体積法の数式を、ハードウェアに近いレベルのコードに分解せずとも、直接的にソースコードとして記述することができる。

4. まとめ

我々は、有限体積法でシミュレーションプログラムを開発する際に、モデルの作成に専念できるように、DSL である有限体積法テンプレートを C++ に埋め込んでいる。この試作版の DSL は、CFD 分野での重要な現象である拡散と対流のうち、拡散現象を記述できるように、作成した。DSL のフロントエンドはモデル作成をサポートするためのオブジェクトを提供し、これらのオブジェクトの機能をバックエンドで意味論モデルが助けている。

この意味論モデルの効用は 2 つある。ひとつは、有限体積法の数式がハードウェアに近い低レベルの C++コード変換するので、開発者が低レベルのコードを書く必要がなくなったこと。もうひとつは、意味論モデルによる数式の変換は、数式テンプレートを介して行なわれるので、一時オブジェクトによる計算速度低下が防止されることである。

現在、この DSL の動作を検証中である。今後は、有限体積法テンプレートを CFD 分野の DSL とするために、意味論モデルが扱える数式の範囲を広げて、また、高速化のために、数式テンプレートの変換の際に並列化やチューニングが合わせて行なわれるようにしていく予定である。

参考文献

[1] M.Fowler, R.Parsons, *Domain-Specific Languages*, Addison-Wesley (2011).

[2] E. Niebler, “Proto: A Compiler Construction Toolkit for DSELs”, ACM

SIGPLAN Symposium on Library-Centric Software Design LCSD'07; http://lcsd.cs.tamu.edu/2007/final/1/1_Paper.pdf ; http://www.boost.org/doc/libs/1_57_0/doc/html/proto/users_guide.html .

[3] H. G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, COMPUTERS IN PHYSICS, VOL. 12, NO. 6, NOV/DEC 1998 ; http://www.openfoam.org/ .

[4] F. Palacios, M. R. Colonno, A. C. Aranake, A. Campos, S. R. Copeland, T. D. Economon, A. K. Lonkar, T. W. Lukaczyk, T. W. R. Taylor, and J. J. Alonso, "Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design", AIAA Paper 2013-0287, 51st AIAA Aerospace Sciences Meeting and Exhibit. January 7th - 10th, 2013. Grapevine, Texas, USA.

[5] T. Veldhuizen, "Expression Templates," C++ Report, Vol. 7 No. 5 June 1995, pp. 26-31; Vandevoorde, David; Josuttis, Nicolai (2002). C++ Templates: The Complete Guide. Addison Wesley. ISBN 0-201-73484-2.

[6] Prud' Homme, C., Chabannes, V., Doyeux, V., Ismail, M., Samake, A., Pena, G.,Trophime, C. (2012). Advances in Feel++: a domain specific embedded language in C++ for partial differential equations. In Eccomas' 12 - European Congress on Computational Methods in Applied Sciences and Engineering. Vienna, Austria.

[7] A. Lani, N. Villedieu, K. Bensassi, L. Kapa, M. Panesi, M. S. Yalim, "COOLFluiD: an open computational platform for multi-physics simulation", 21st AIAA CFD Conference, AIAA 2013-2589, San Diego, Jun 2013.

[8] J. Gratien, , CppNow 2012.

[9] Pierre Esterie, Joel Falcou, Mathias Gaunard, Jean-Thierry Lapreste, Lionel Lacassagne, Journal of Parallel and Distributed Computing, Volume 74, Issue 12, December 2014, Pages 3240-3253 .

[10] H.K.Versteeg, W. Malalasekera, *An Introduction of Computational Fluid Dynamics – The Finite Volume Method 2nd Ed.*, Pearson Educational Limited (2007); 松下洋介ら（共訳）: 数値流体力学 第 2 版, 森北出版株式会社 (2011).